接口

学习目标

- 理解接口的概念
- 学会通过接口标注复杂结构的对象

接口定义

前面我们说到，TypeScript 的核心之一就是对值（数据）所具有的结构进行类型检查，除了一些前面说到基本类型标注，针对对象类型的数据，除了前面提到的一些方式意外，我们还可以通过： Interface（接口），来进行标注。

接口：对复杂的对象类型进行标注的一种方式，或者给其它代码定义一种契约（比如：类）

接口的基础语法定义结构特别简单

```
interface Point {
    x: number;
    y: number;
}
```

上面的代码定义了一个类型，该类型包含两个属性，一个 number 类型的 x 和一个 number 类型的 y，接口中多个属性之间可以使用 逗号 或者 分号 进行分隔

我们可以通过这个接口来给一个数据进行类型标注

```
let p1: Point = {
    x: 100,
    y: 100
};
```

> 注意：接口是一种 类型 ，不能作为 值 使用

```
interface Point {
    x: number;
    y: number;
}

let p1 = Point; //错误
```

当然，接口的定义规则远远不止这些

可选属性

接口也可以定义可选的属性，通过 ? 来进行标注

```
interface Point {
    x: number;
    y: number;
    color?: string;
}
```

其中的 color? 表示该属性是可选的

只读属性

我们还可以通过 readonly 来标注属性为只读

```
interface Point {
    readonly x: number;
    readonly y: number;
}
```

当我们标注了一个属性为只读，那么该属性除了初始化以外，是不能被再次赋值的

任意属性

有的时候，我们希望给一个接口添加任意属性，可以通过索引类型来实现

数字类型索引

```
interface Point {
    x: number;
    y: number;
    [prop: number]: number;
}
```

字符串类型索引

```
interface Point {
    x: number;
    y: number;
    [prop: string]: number;
}
```

数字索引是字符串索引的子类型

> 注意：索引签名参数类型必须为 string 或 number 之一，但两者可同时出现

```
interface Point {
    [prop1: string]: string;
    [prop2: number]: string;
}
```

> 注意：当同时存在数字类型索引和字符串类型索引的时候，数字类型的值类型必须是字符串类型的值类型或子类型

```
interface Point1 {
    [prop1: string]: string;
    [prop2: number]: number;    // 错误
}
interface Point2 {
    [prop1: string]: Object;
    [prop2: number]: Date;  // 正确
}
```

使用接口描述函数

我们还可以使用接口来描述一个函数

```
interface IFunc {
  (a: string): string;
}

let fn: IFunc = function(a) {}
```

> 注意，如果使用接口来单独描述一个函数，是没 `key` 的

接口合并

多个同名的接口合并成一个接口

```
interface Box {
    height: number;
    width: number;
}

interface Box {
    scale: number;
}

let box: Box = {height: 5, width: 6, scale: 10}
```

- 如果合并的接口存在同名的非函数成员，则必须保证他们类型一致，否则编译报错
- 接口中的同名函数则是采用重载（具体后期函数详解中讲解）